On the provable security of BEAR and LION schemes

Lara Maines (lara.maines@gmail.com)
Department of Mathematics, University of Trento, Italy

Matteo Piva (matteo.piva@unitn.it)
Department of Mathematics, University of Trento, Italy

Anna Rimoldi (anna.rimoldi@gmail.com)
eRISCS, Universite de la Méditerranée, Marseille, France

Massimiliano Sala (maxsalacodes@gmail.com)
Department of Mathematics, University of Trento, Italy

Abstract

BEAR, LION and LIONESS are block ciphers presented by Biham and Anderson (1996), inspired by the famous Luby-Rackoff constructions of block ciphers from other cryptographic primitives (1988). The ciphers proposed by Biham and Anderson are based on one stream cipher and one hash function. Good properties of the primitives ensure good properties of the block cipher. In particular, they are able to prove that their ciphers are immune to any efficient known-plaintext key-recovery attack that can use as input only one plaintext-ciphertext pair. Our contribution is showing that these ciphers are actually immune to any efficient known-plaintext key-recovery attack that can use as input any number of plaintext-ciphertext pairs. We are able to get this improvement by using slightly weaker hypotheses on the primitives. We also discuss the attack by Morin (1996).

Keywords: Cryptography, block cipher, stream cipher, hash function, BEAR, LION, Luby-Rackoff cipher.

Introduction

In this paper we discuss three block ciphers, BEAR, LION and LIONESS, proposed in [AB96] by Anderson and Biham, whose construction depends on one stream cipher and one hash function. These block ciphers are inspired by [LR88] and present a three-round (for BEAR and LION) or four-round (for

LIONESS) Feistel construction. In particular, we treat the provable security shown by them and provide some improvements.

In Section 1 we give some preliminaries, recalling in particular BEAR's construction (Subsection 1.1) with the results by Anderson and Biham, Th. 1.1 and Th. 1.2, that ensure the non-existence of efficient attacks (of a very specific kind) on BEAR if at least one of the two primitives is robust. In this section we also recall LION's construction (Subsection 1.2) and their claimed results on LION, Th. 1.6 and Th. 1.8, on the non-existence of similar attacks. We provide our proof for them, with slightly weaker hypotheses. This preliminary section is concluded by a description of LIONESS (Subsection 1.3).
In Section 2 we give our results on BEAR, LION and LIONESS, that show the non-existence of some more general attacks. We also introduce two slight variations, BEAR2 and LION2, of BEAR and LION, respectively. We identify the hypotheses on the primitives that we need, in particular highlighting the relation between key and hash function in the keyed hash function. In Subsection 2.1 we provide Th. 2.4 that improves Th. 1.2, and Th. 2.7 on BEAR2, that improves Th. 1.1. In Subsection 2.2 we provide Th. 2.9 that improves Th. 1.6, and Th. 2.10 on LION2, that extends and improves Th. 1.8. Finally, in Subsection 2.3 we extend our results to LIONESS in Th. 2.13 and Th. 2.14. In Section 3, we discuss our results and draw our conclusions. We also put in context the attack to BEAR and LION by Morin ([Mor96]).

1 Preliminaries

We use $\mathbb{F}$ to denote $\mathbb{F}_2$ and typically when a capital R or a capital L appear, they mean elements of $\mathbb{F}^r$ and $\mathbb{F}^l$ respectively, with $r > l$. The encrypted/decrypted messages are of kind $(L_i, R_i) \in \mathbb{F}^{l+r}$. The key space is denoted by $\mathcal{K}$. Usually the key $K = (K_1, K_2)$ is composed of two subkeys, each of length greater than l, so $\mathcal{K} = \mathbb{F}^k \times \mathbb{F}^k$, $k \geq l$.

In this paper we consider oracles able to recover the key using as input only a set of known plaintexts/ciphertexts. We call "single-pair" any oracle so strong as to need only one pair and "multi-pair" any other.

1.1 Preliminaries on BEAR

The description of BEAR encryption/decryption is based on a keyed hash function H_K and a stream cipher S with the following properties.

(1) The keyed hash function $H_K(M)$
   (a) is based on an unkeyed hash function $H'(M)$, in which we append and/or prepend the key to the message;
   (b) is one-way and collision-free, i.e. it is hard given Y to find X such that $H'(X) = Y$, and to find unequal X and Y such that $H'(X) = H'(Y)$;
   (c) is pseudo-random, in that even given $H'(X_i)$ for any set of inputs, it is hard to predict any bit of $H'(Y)$ for a new input Y.

(2) The stream cipher $S(M)$:

- (0) is pseudo-random (this condition is assumed but not listed in [AB96]);
- (a) resists key recovery attacks, in that it is hard to find the seed X given $Y = S(X)$;
- (b) resists expansion attacks, in that it is hard to expand any partial stream of Y.

We note that conditions (1)-c and (2)-0 ensure respectively that H_K and S are pseudo-random, in order to obtain security against some distinguishing attacks in a rather theoretical model ([LR88] and [Luc96]).

We recall the BEAR encryption/decryption scheme (here $k > l$).

ENCRYPTION	DECRYPTION
$\overline{L} = L + H_{K_1}(R)$	$\overline{L} = L' + H_{K_2}(R')$
$R' = R + S(\overline{L})$	$R = R' + S(\overline{L})$
$L' = \overline{L} + H_{K_2}(R')$	$L = \overline{L} + H_{K_1}(R)$

In [AB96] Anderson and Biham claim the following results on one-pair oracles.

Theorem 1.1 (Th. 1 of [AB96])**.** *An oracle which finds the key of BEAR, given one plaintext/ciphertext pair, can efficiently and with high probability find the seed M of the stream cipher S for any output $Y = S(M)$.*

Theorem 1.2 (Th. 2 of [AB96])**.** *An oracle which finds the key of BEAR, given one plaintext/ciphertext pair, can efficiently and with high probability find preimages and collisions of the hash function H.*

Remark 1.3. We observe that while proving Th. 1.1 and Th. 1.2 they **only** need the following assumption on H:

for most R's the map $H^R : \mathbb{F}^k \mapsto \mathbb{F}^l$, $H^R(K) = H_K(R)$, is surjective.

This assumption is implied by the pseudo-randomness of the unkeyed hash function H' (and the bigger dimension of the key space), but it is not equivalent to it. Indeed, it is easy to construct even linear functions satisfying it.
On the other hand, **no** hypothesis on the stream cipher S is used.
Remark 1.4. The word `efficiently` in Th. 1.1 and Th. 1.2 might be confusing. The oracle could need huge resources to work. A trivial example is given by a brute force search of all keys. What Biham and Anderson mean is that the attacker will need little computational effort `in addition to` any effort done by the oracle itself, whatever large.

As a direct consequence of Th. 1.1 and Th. 1.2 we have:

Corollary 1.5. *If it is impossible to find efficiently the seed of S or it is impossible to find efficiently preimages and collisions of H, then no efficient (key-recovery) single-pair attack exists for BEAR.*

1.2 Preliminaries on LION

LION is quite similar to BEAR except that it uses the stream cipher twice and the hash function only once. For LION, $\mathcal{K} = \mathbb{F}^{2l}$.
The requests are:

(1) The hash function $H(M)$:

  (b) is one-way and collision-free, i.e. it is hard given Y to find X such that $H'(X) = Y$, and to find unequal X and Y such that $H'(X) = H'(Y)$;

(2) The stream cipher $S(M)$:

  (0) is pseudo-random,
  (a) resists key recovery attacks, in that it is hard to find the seed X given $Y = S(X)$;
  (b) resists expansion attacks, in that it is hard to expand any partial stream of Y.

We note that Anderson and Biham here dropped 1-(a) and 1-(c).

We recall the LION encryption/decryption scheme (here $k = l$).

ENCRYPTION	DECRYPTION
$\overline{R} = R + S(L + K_1)$	$\overline{R} = R' + S(L' + K_2)$
$L' = L + H(\overline{R})$	$L = L' + H(\overline{R})$
$R' = \overline{R} + S(L' + K_2)$	$R = \overline{R} + S(L + K_1)$

Results similar to Theorem 1.1 and Theorem 1.2 are claimed also for LION but without proof nor precise statement. They write *the security reduction of LION proceeds similarly to that of BEAR; an oracle which yields the key of LION will break both its components.*. Unfortunately, we have not been able to write down direct adaptions of the previous proofs, especially because here the property of H as in Remark 1.3 cannot be used and we do not see how one could get something similar for the stream cipher. Therefore, we now state precisely their claims, giving the weakest hypotheses we can exhibit.

Theorem 1.6. *Assume nothing on H and S, except that they are set functions $H : \mathbb{F}^r \to \mathbb{F}^l$ and $S : \mathbb{F}^l \to \mathbb{F}^r$. An oracle $\mathcal{A}_1$ which finds the key of LION, given one plaintext/ciphertext pair, can efficiently and with high probability find the seed M of the stream cipher S for any particular output $Y = S(M)$.*

Proof. Let us choose a random input (L, R). Let $K_1 = M + L$. Then $S(L + K_1) = Y$. We can compute $\overline{R} = R + Y$, $L' = L + H(R + Y)$ and, by choosing any K_2, $R' = R + Y + S(L' + K_2)$. Then we give in input to the oracle the pair $\{ (L, R), (L', R') \}$ and $\mathcal{A}_1$ returns (K_1, K_2), so we can immediately compute $M = L + K_1$. □

To prove a similar theorem for the hash function, we need the following definition.

Definition 1.7. *Let H and S be functions, $H : \mathbb{F}^r \to \mathbb{F}^l$ and $S : \mathbb{F}^l \to \mathbb{F}^r$, with $r \geq l$. We say that (S, H) is a* good pairing *if for a random $Y \in \mathbb{F}^l$ we have $H^{-1}(Y) \cap \mathrm{Im}(S) \neq \emptyset$.*

We note that if at least one between H or S is pseudo-random, then (S, H) is a good pairing. However, we might have a good pairing even if none of the primitives is pseudo-random.

We are ready for our interpretation of their claim on the link between the security of LION and of the hash function.

Theorem 1.8. *Assume that (S, H) is a good pairing. An oracle $\mathcal{A}_1$ which finds the key of LION, given one plaintext/ciphertext pair, can efficiently and with high probability find preimages and collisions of the hash function H.*

Proof. Since $r > l$ we can choose $\tilde{R} \notin \mathrm{Im}(S)$ with probability $\frac{2^r - 2^l}{2^r}$ and calculate $H(\tilde{R}) = \tilde{Y} \in \mathbb{F}^l$. We can suppose $H^{-1}(\tilde{Y}) \cap \mathrm{Im}(S) \neq \emptyset$ (else we can choose another $\tilde{R}$) and so there is an $X \in H^{-1}(\tilde{Y}) \cap \mathrm{Im}(S)$. We consider as plaintext $(L, 0)$, where L is any element of $\mathbb{F}^l$ and $0 \in \mathbb{F}^r$. There exists K_1 such that $\overline{R} = S(L + K_1) = X$, because $X \in \mathrm{Im}(S)$. Thus $L' = L + H(\overline{R}) = L + H(X) = L + \tilde{Y}$, because $X \in H^{-1}(\tilde{Y})$. It follows that for $K_2 = L + \tilde{Y} + X$ we have $R' = X + S(L' + K_2) = X + X = 0$. We give to $\mathcal{A}_1$ as input the pair $\left\{ (L, 0), (L + \tilde{Y}, 0) \right\}$ and it returns (K_1, K_2), so we can compute easily $X = S(L + K_1)$, finding a collision $H(\tilde{R}) = H(X) = \tilde{Y}$. Note that $\tilde{R} \neq X$, since $\tilde{R} \notin \mathrm{Im}(S)$ and $X \in \mathrm{Im}(S)$.

To find a preimage, argue as above but with an arbitrary $Y \in \mathbb{F}^l$. □

The same considerations as in Remark 1.4 hold and a corollary analogous to Cor. 1.5 holds.

1.3 Preliminaries on LIONESS

The third block cipher proposed in [AB96] is LIONESS, which consists of four rounds and uses four independent keys, K_1, $K_3 \in \mathbb{F}^l$, K_2, $K_4 \in \mathbb{F}^k$, so $\mathcal{K} = \mathbb{F}^l \times \mathbb{F}^k \times \mathbb{F}^l \times \mathbb{F}^k$, for some k.

Anderson and Biham do not give explicit statements on LIONESS's security, but it is obvious from its construction that any provable-security result for LION and/or BEAR directly extends to LIONESS, because any oracle attacking LIONESS will be able to attack LION and BEAR, with possibly even less effort.

LIONESS

ENCRYPTION	DECRYPTION
$\overline{R} = R + S(L + K_1)$	$\overline{L} = L' + H_{K_4}(R')$
$\overline{L} = L + H_{K_2}(\overline{R})$	$\overline{R} = R' + S(\overline{L} + K_3)$
$R' = R + S(\overline{L} + K_3)$	$L = \overline{L} + H_{K_2}(\overline{R})$
$L' = \overline{L} + H_{K_4}(R')$	$R = \overline{R} + S(L + K_1)$

For completeness, we can state the following obvious corollary.

Corollary 1.9. *Assume nothing on H and S, except that they are set functions $H : \mathbb{F}^r \to \mathbb{F}^l$ and $S : \mathbb{F}^l \to \mathbb{F}^r$ and that for most R's the map $H^R : \mathbb{F}^k \mapsto \mathbb{F}^l$, $H^R(K) = H_K(R)$, is surjective. Then an oracle $\mathcal{A}_1$ which finds the key of LIONESS, given one plaintext/ciphertext pair, can efficiently and with high probability both find the seed of S and find preimages/collisions of H.*

2 Our improvements

We propose a property for the keyed hash function, H_K, able to ensure the security from any key-recovery attack that uses plaintext/ciphertext pairs.

Definition 2.1. *Given a keyed hash function $\mathcal{H} = \{ H_K \}_{K \in \mathbb{F}^k}$, $H_K : \mathbb{F}^r \mapsto \mathbb{F}^l$ for any $K \in \mathbb{F}^k$, we say that $\mathcal{H}$ is* key-resistant *if, given a pair (Z, R) such that $Z = H_K(R)$ for a random K and a random R, then it is hard to find K.*

Let us consider a keyed hash function $\mathcal{H}$ of kind $H_K = H'(f(K, R))$ for some injective function f. For practical purposes we want also that (K, R) is easy to find from $f(K, R)$. In [AB96] f can be a concatenation, but we do not need to be so restrictive. Let K and R be random and consider the equation:

$$Z = H'(f(K, R)) = H_K(R) \tag{1}$$

We note that 1-(b) for H' *implies* that (1) cannot be solved knowing Z. On the other hand, the key-resistance of $\mathcal{H}$ *means* that (1) cannot be solved knowing Z *and* R. It may seem that there is a logical link between the two conditions, but generally speaking there is none, as we are going to show:

- Suppose that 1-(b) does not hold. From K and R we get Z. Then we can solve $Z = H'(X)$. However, Z can have many preimages (2^{r-l} on average), and so X is likely to be outside $\mathrm{Im}(f)$. The knowledge of R cannot help here, except in discarding unwanted preimages. Only if 1-(b) fails badly, that is, if we can get efficiently all preimages of Z, then we will be able to solve (1) by discarding all preimages except that of the desired form (K, R).

- Suppose that $\mathcal{H}$ is not key-resistant. If Z does not come from $\mathrm{Im}(f)$ then there is no way the lack of key-resistance can help. But even if $Z = H(K, R)$ for some random K and R, still the attacker does not know R and so lack of key-resistance cannot help, unless the attacker is allowed to search in a brute-force effort the whole $\mathbb{F}^r$, which is supposed to be hard in our context.

If H' is pseudo-random, then $\mathcal{H}$ is clearly key-resistant.

As regards the stream-cipher, we can consider a similar notion in a slightly more general situation, that is, when $\mathcal{K} \subset \mathbb{F}^{2l}$.

Definition 2.2. *Let $\mathcal{K} \subset \mathbb{F}^{2l}$. Given a stream cipher $S : \mathbb{F}^l \mapsto \mathbb{F}^r$, we say that S is* key-resistant *if, given a pair (Z, L) such that $Z = S(L + K_1)$ for a random $(K_1, K_2) \in \mathcal{K}$ and a random $L \in \mathbb{F}^l$, then it is hard to find K_1.*

When $\mathcal{K} = \mathbb{F}^{2l}$ we obviously have the equivalence between 2-(a) and the key-resistance, since translations act regularly.

Remark 2.3. We could change the definition of LION by having a different action induced by the keys, that is, $S(\tau_K(L))$ instead of $S(L + K)$, where $\{\tau_K\}_{K\in\mathcal{K}} \subset \mathrm{Sym}(\mathbb{F}^l)$, $\mathrm{Sym}(\mathbb{F}^l)$ being the symmetric group acting on $\mathbb{F}^l$. All subsequent results will still hold, provided the action is regular.

2.1 Our improvements for BEAR

It is possible to give an improvement of Theorem 1.2, passing from one-pair oracles to multi-pair oracles.

Theorem 2.4. *Let $n \geq 1$. Let $\mathcal{A}_n$ be an oracle able to find the key of BEAR given any set of n plaintext-ciphertext pairs $\{\, ((L_i, R_i), (L'_i, R'_i)) \,\}_{1\leq i\leq n}$. Then $\mathcal{A}_n$ is able to solve efficiently any equation $Z = H_{K_1}(R)$, knowing Z and R, for any random $R \in \mathbb{F}^r$ and any random $K_1 \in \mathbb{F}^k$.*

Proof. Let us choose a set $\{\, L_i \,\}_{1\leq i\leq n} \subset \mathbb{F}^l$ and consider the set of plaintexts $\{\, (L_i, R) \,\}_{1\leq i\leq n}$. It is possible to generate a set of ciphertexts $\{\, (L'_i, R'_i) \,\}_{1\leq i\leq n}$ by choosing any sub-key K_2 and computing: $\overline{L}_i = L_i + Z$, $R'_i = R + S(L_i + Z)$, $L'_i = L_i + Z + H_{K_2}(R'_i)$. With $\{\, ((L_i, R), (L'_i, R'_i)) \,\}_{1\leq i\leq n}$ as input, $\mathcal{A}_n$ outputs K_2, which was already known, and K_1, which was unknown. □

Again, when we say *efficiently* we disregard any effort put by the oracle itself (see Remark 1.4).

Corollary 2.5. *If the (keyed) hash function is key-resistant, no efficient multi-pair oracle exists for BEAR.*

Unfortunately we have not been able to obtain a direct improvement of Theorem 1.1, but it is quite simple to modify BEAR in order to obtain a similar result also for the stream cipher. Let us consider the following variation

of BEAR's scheme, in which $\mathcal{K} \subset \mathbb{F}^k \times \mathbb{F}^l \times \mathbb{F}^k$, for some k, with K_1, $K_3 \in \mathbb{F}^k$ and $K_2 \in \mathbb{F}^l$.

BEAR 2

ENCRYPTION	DECRYPTION
$\overline{L} = L + H_{K_1}(R)$	$\overline{L} = L' + H_{K_3}(R')$
$R' = R + S(\overline{L} + K_2)$	$R = R' + S(\overline{L} + K_2)$
$L' = \overline{L} + H_{K_3}(R')$	$L = \overline{L} + H_{K_1}(R)$

First we extend Th. 2.4 from BEAR to BEAR2.

Theorem 2.6. *Let $n \geq 1$. Let $\mathcal{A}_n$ be an oracle able to find the key of BEAR 2 given any set of n plaintext-ciphertext pairs $\{ ((L_i, R_i), (L'_i, R'_i)) \}_{1 \leq i \leq n}$. Then $\mathcal{A}_n$ is able to solve any equation $Z = H_{K_1}(R)$, knowing Z and R, for any random $R \in \mathbb{F}^r$ and any random $K_1 \in \mathbb{F}^k$.*

Proof. Obvious adaption of the proof of Th. 2.4. We choose this time K_2 and K_3, we obtain K_1 again. □

Now we are ready for the following result, linking the security of BEAR2 also to the properties of the stream cipher S, in a multi-pair context.

Theorem 2.7. *Let $n \geq 1$. Let $\mathcal{A}_n$ be an oracle able to find the key of BEAR2 given any set of n plaintext-ciphertext pairs $\{ ((L_i, R_i), (L'_i, R'_i)) \}_{1 \leq i \leq n}$. Then $\mathcal{A}_n$ is able to solve any equation $Z = S(X + K_2)$, knowing Z and X, for any random $X \in \mathbb{F}^l$ and any random $K_2 \in \mathbb{F}^l$.*

Proof. Let us choose a set $\{ R_i \}_{1 \leq i \leq n} \subset \mathbb{F}^r$ and two sub-keys K_1, K_3. It is possible to generate plaintext/ciphertext pairs by choosing $L_i = X + H_{K_1}(R_i)$ and computing: $\overline{L_i} = L_i + H_{K_1}(R_i) = X$, $R'_i = R_i + Z$, $L'_i = X + H_{K_3}(R'_i)$. We give in input to $\mathcal{A}_n$ the set $\{ (L_i, R_i), (L'_i, R'_i) \}_{1 \leq i \leq n}$, $\mathcal{A}_n$ returns K_1, K_3 which were already known, and K_2, which was unknown. □

We can summarize our findings on BEAR2 in the following corollary.

Corollary 2.8. *No efficient multi-pair key-recovery oracle exists for BEAR2 if the hash function is key-resistant or the stream cipher is key-resistant.*

2.2 Our improvements for LION

A result similar to Theorem 2.4 holds for LION.

Theorem 2.9. *Let $n \geq 1$. Let $\mathcal{A}_n$ be an oracle able to find the key of LION given any set of n plaintext-ciphertext pairs $\{ ((L_i, R_i), (L'_i, R'_i)) \}_{1 \leq i \leq n}$. Then $\mathcal{A}_n$ is able to solve any equation $Z = S(L + K_1)$, knowing Z and L, for any random $L \in \mathbb{F}^l$ and any random $K_1 \in \mathbb{F}^k$.*

Proof. Let us choose a set $\{ R_i \}_{1 \leq i \leq n} \subset \mathbb{F}^r$ and consider the set of plaintexts $\{ (L, R_i) \}_{1 \leq i \leq n}$. It is possible to generate a set of ciphertexts $\{ (L'_i, R'_i) \}_{1 \leq i \leq n}$ by choosing any sub-key K_2 and computing: $\overline{R}_i = R_i + S(L + K_1) = R_i + Z$, $L'_i = L_i + H(R_i + Z)$, $R'_i = R_i + Z + S(L'_i + K_2)$. Using $\mathcal{A}_n$ we can find K_2, which was already known, and K_1, which was unknown. □

As we have already seen for BEAR, we have not been able to extend a result similar to Theorem 2.9 also for its hash functions, but it is quite simple to modify LION in order to obtain it, as in the following table, where K_1, $K_3 \in \mathbb{F}^l$ and $K_2 \in \mathbb{F}^k$, and so $\mathcal{K} \subset \mathbb{F}^l \times \mathbb{F}^k \times \mathbb{F}^l$ for some k.

LION2

ENCRYPTION	DECRYPTION
$\overline{R} = R + S(L + K_1)$	$\overline{R} = R' + S(L' + K_3)$
$L' = L + H_{K_2}(\overline{R})$	$L = L' + H_{K_2}(\overline{R})$
$R' = \overline{R} + S(L' + K_3)$	$R = \overline{R} + S(L + K_1)$

Theorem 2.10. *Let $n \geq 1$. Let $\mathcal{A}_n$ be an oracle able to find the key of LION2 given any set of n plaintext-ciphertext pairs $\{ ((L_i, R_i), (L'_i, R'_i)) \}_{1 \leq i \leq n}$. Then $\mathcal{A}_n$ is able to solve any equation $Z = S(L + K_1)$, knowing Z and L, for any random $L \in \mathbb{F}^l$ and any random $K_1 \in \mathbb{F}^k$.*

Proof. Obvious adaption of the proof of Th. 2.9. □

Theorem 2.11. *Let $n \geq 1$. Let $\mathcal{A}_n$ be an oracle able to find the key of LION 2 given any set of n plaintext-ciphertext pairs $\{ ((L_i, R_i), (L'_i, R'_i)) \}_{1 \leq i \leq n}$. Then $\mathcal{A}_n$ is able to solve any equation $Z = H_{K_2}(X)$, knowing Z and X, for any random $X \in \mathbb{F}^r$ and any random $K_2 \in \mathbb{F}^k$.*

Proof. Let us choose a set $\{ L_i \}_{1 \leq i \leq n} \subset \mathbb{F}^l$ and any sub-keys K_1, $K_3 \in \mathbb{F}^l$. It is possible to generate plaintext/ciphertext pairs by choosing $R_i = X + S(L_i + K_1)$ and computing: $\overline{R}_i = R_i + S(L_i + K_1) = X + S(L_i + K_1) + S(L_i + K_1) = X$, $L'_i = L_i + Z$, $R'_i = X + S(L'_i + K_3)$. We give in input to $\mathcal{A}_n$ the set $\{ (L_i, R_i), (L'_i, R'_i) \}$, $\mathcal{A}_n$ returns K_1, K_3, which were already known, and K_2, which was unknown. □

We can summarize our findings on LION and LION2 in the following corollary.

Corollary 2.12. *No efficient multi-pair key-recovery oracle exists for LION if the stream cipher is key-resistant.*

No efficient multi-pair key-recovery oracle exists for LION2 if the hash function is key-resistant or the stream cipher is key-resistant.

2.3 Our improvements for LIONESS

Since LIONESS combines the construction of LION and BEAR, it is quite obvious that any provable-security result holding for BEAR and LION still holds for LIONESS. For completeness, we give the formal proofs for our multi-pair results.

Theorem 2.13. *Let $n \geq 1$. Let $\mathcal{A}_n$ be an oracle able to find the key of LIONESS given any set of n plaintext-ciphertext pairs $\{ ((L_i, R_i), (L'_i, R'_i)) \}_{1 \leq i \leq n}$. Then $\mathcal{A}_n$ is able to solve any equation $Z = S(L + K_1)$, knowing Z and L, for any random $L \in \mathbb{F}^l$ and any random $K_1 \in \mathbb{F}^l$.*

Proof. Let us choose a set $\{ R_i \}_{1 \leq i \leq n} \subset \mathbb{F}^r$ and consider the set of plaintexts $\{ (L, R_i) \}_{1 \leq i \leq n}$. It is possible to generate a set of ciphertexts $\{ (L'_i, R'_i) \}_{1 \leq i \leq n}$ by choosing any sub-keys K_2, K_3, K_4 and computing: $\overline{R}_i = R_i + Z$, $\overline{L}_i = L + H_{K_2}(\overline{R}_i)$, $R'_i = R_i + S(\overline{L}_i + K_3)$ and $L' = \overline{L}_i + H_{K_4}(R'_i)$. Using $\mathcal{A}_n$ we can find K_2, K_3, K_4, which were already known, and K_1, which was unknown. □

Theorem 2.14. *Let $n \geq 1$. Let $\mathcal{A}_n$ be an oracle able to find the key of LIONESS given any set of n plaintext-ciphertext pairs $\{ ((L_i, R_i), (L'_i, R'_i)) \}_{1 \leq i \leq n}$. Then $\mathcal{A}_n$ is able to solve any equation $Z = H_{K_4}(R')$, knowing Z and R', for any random $R' \in \mathbb{F}^r$ and any random $K \in \mathbb{F}^k$.*

Proof. Let us choose a set $\{ L_i \}_{1 \leq i \leq n} \subset \mathbb{F}^l$ and consider the set of ciphertexts $\{ (L'_i, R') \}_{1 \leq i \leq n}$. It is possible to generate a set of plaintexts $\{ (L_i, R_i) \}_{1 \leq i \leq n}$ by choosing any sub-keys K_1, K_2, K_3 and decrypting: $\overline{L_i} = L'_i + Z$, $\overline{R_i} = R' + S(\overline{L_i} + K_3)$, $L_i = \overline{L_i} + H_{K_2}(\overline{R_i})$, $R_i = \overline{R_i} + S(L_i + K_1)$. Using $\mathcal{A}_n$ we can find K_1, K_2, K_3, which were already known, and K_4, which was unknown. □

3 Conclusions and further comments

Let us consider a keyed hash function with a very weak requirement, i.e., that it is surjective both fixing the key and with respect to the keys (see Remark 1.3). Anderson and Biham prove that no single-pair oracle exists for BEAR, under the assumption that "the stream seed is difficult to recover OR the hash function is collision resistant OR the hash preimage is hard to recover". We prove that no multi-pair oracle exists for BEAR under the assumption that "hash is key-resistant". We also suggest a slight modification of BEAR, BEAR 2, where we can prove that no multi-pair oracle exists under the assumption that "hash is key-resistant OR stream is key-resistant".

The conclusions about key-recovery attacks for LION are quite similar to those for BEAR. Anderson and Biham claim without proof that no single-pair oracle exists for LION under the assumption that "the stream seed is difficult to recover OR the hash function is collision resistant OR the hash preimage is hard to recover". However, we have found no direct proof following

their outline. We prove that no single-pair oracle exists for LION under the assumptions that the stream seed is difficult to recover. To prove the same thing with assumptions on the hash function, we need a condition that we call *good pairing*. Interestingly, this condition follows from the pseudo-random nature of S OR the pseudo-random nature of H. Given the good pairing for granted, we finish to prove their claim, that is, no single-pair oracle exists for LION under the assumption that "the hash function is collision resistant OR the hash preimage is hard to recover". As in the case of BEAR, we prove that no multi-pair oracle exists for LION under the assumption that "the stream cipher is key-resistant", which is equivalent to "the stream preimage is hard to recover" in many pratical situations. We also suggest a slight modification of LION, LION 2, where we can prove that no multi-pair oracles exist under the assumption that "the hash function is key-resistant OR the stream cipher is key-resistant".

As regards key-recovery attacks, LIONESS's virtues are the sum of LION's and BEAR's virtues. So it is possible to prove the non-existence of one-pair oracles using the authors' assumptions, but we can indeed prove the non-existence of multi-pair oracles under only the key-resistance assumption.

We note that an attack by Morin ([Mor96]) has somehow diminuished the confidence in the robustness of these schemes. However, the attack succeeds only because its brute force search on the round function contradicts the key-resistance of the hash function and of the stream function. So, whenever $\mathcal{H}$ or S remain key-resistant, both LION and BEAR are immune to such attacks.

Acknowledgements

For their comments and suggestions the authors would like to thank E. Bellini, G. Morgari and M. Coppola. The first three authors would like to thank their supervisor (the fourth author).

This work has been supported by TELSY Elettronica e Telecomunicazioni, an Italian company working in Information and Communication Security.

References

[AB96] R. Anderson and E. Biham, *Two practical and provably secure block ciphers: BEAR and LION*, Proc. of FSE 1996, LNCS, vol. 1039, 1996, pp. 113–120.

[LR88] M. Luby and C. Rackoff, *How to construct pseudorandom permutations from pseudorandom functions*, SIAM J. Comput. **17** (1988), no. 2, 373–386.

[Luc96] S. Lucks, *Faster Luby-Rackoff Ciphers*, Proc. of FSE 1996, LNCS, vol. 1039, 1996, pp. 189–203.

[Mor96] P. Morin, *Provably secure and efficient block ciphers*, Proc. of SAC 1996, 1996, pp. 30–37.